MHaptic : a Haptic Manipulation Library for Generic Virtual Environments

Renaud Ott, Vincent De Perrot, Daniel Thalmann and Frédéric Vexo
Virtual Reality Laboratory (VRLab)
École Polytechnique Fédérale de Lausanne (EPFL)
1015 Lausanne, Switzerland
Telephone: (+41) 21–693–5215
Fax: (+41) 21–693–5328
Email: {renaud.ott, vincent.deperrot, daniel.thalmann, frederic.vexo}@epfl.ch

***Abstract*—This paper presents a new library called MHaptic for bimanual haptic interaction within generic virtual environments. It has been specifically designed to work with a Haptic Workstation™. MHaptic provides tools for accelerated development of virtual environment applications with haptic feedback like device calibration, user comfort improvements and access to low level parameters. Due to its integration with the Ageia PhysX library, it facilitates the dynamic animation of virtual objects. A realistic hand model based on mass-spring systems allows natural and intuitive manipulation. MHaptic is complemented by an authoring tool that associates information required for haptic feedback to existing virtual environments. The combination of the library and the authoring tool creates a framework for easy development of complex VR haptic applications.**

I. Introduction

Virtual Reality refers to the technology that aims at immersing a subject into different environments without physically moving him. Most of the VR systems includes visual and auditive experiences, as they are easily reproduced through popular devices like screens and speakers. In order to increase immersion and interaction, advanced VR systems includes also haptic devices to simulate the sense of touch and proprioception. This could not be achieved by using the classic keyboard and mouse interface.

In this paper, we focus on the proprioception by the mean of a Haptic Workstation™, a haptic peripheral that has the unique advantage to be a two-handed device, allowing for interacting by a very intuitive and natural manner (see figure 1). Because none of the existing libraries were intended to this specific purpose, it was hard to reuse them. Thus it forces us to develop a new one managing the Haptic Workstation™, and allowing a user to touch/grasp/manipulate any kind of virtual environment using his hands. Moreover, we notice the lack of haptic information included in existing Virtual Environments, so we propose also an application allowing to quickly augment visual 3D models in order to include haptic data.

In this paper, we will first present the related work, by giving a brief overview of the existing haptic libraries, justifying the creation of our library. Then, in the third section, we will go into the description of the haptic library, i.e. the structure, the functionalities, the main haptic algorithms and techniques developed to ensure a convincing immersion. In the fourth section, we introduce a software, the Haptic Scene Creator (HSC), which is also part of the MHaptic framework, that we developed to allow manipulation in any Virtual Environments. Finally, the last section provides the results obtained when using this library, and concludes this paper.

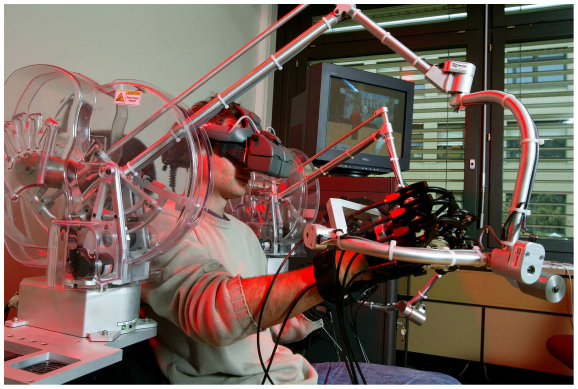

Fig. 1. The Immersion Haptic Workstation™.

II. A new Haptic Library

Contrary to 3D computer graphics, standard haptic libraries that manage and take advantage of all the haptic devices does not really exist. In computer graphics, Direct3D or OpenGL allows for simplifying the creation of visual applications, whatever the graphic card. This is mainly due to the technical similarities in the pipeline for displaying visual content. However, haptic rendering could be conveyed by many strongly different devices. In this section, we will present the main haptic rendering software and the associated device, and finally, list our needs according to this specific haptic device.

A. State of the Art in Haptic libraries

Most of the haptic libraries are dedicated to one device. Usually, they are developed for a specific hardware by the company providing it. For example, Microsoft designed DirectInput for supporting their SideWinder Joystick.

Fig. 2. The four devices of the Haptic Workstation™. On the left, the CyberGrasp™ and CyberGlove®. On the right, the CyberForce® and CyberTrack™.

Another example is SensAble Technologies which has created the GHOST® SDK. This library supports only the Phantom® device product line, and eases the creation of applications that use it. Today, it has been replaced by the OpenHaptics™ Toolkit, but this new one is still dedicated to the Phantom® devices. A last example comes with the Force Dimension API that is a low-level API providing control over the Force Dimension products.

Considering all these hardware-dependant libraries, some researches have been done for creating software that provides access to many kind of devices, trying to abstract the device itself and focus on the haptic rendering algorithm. We can cite two of these libraries: ReachIn[1] and Chai3D[2]. The first one is a complete framework allowing the creation of Virtual Environments that combines haptic, audio and visual feedback. Supported hardware devices are the Phantom, the Force Dimension Delta, the new Novint Falcon low-cost products, and some others. However, ReachIn is not open source and thus, can not be extended to support a specific device like the Haptic Workstation. The other library previously mentioned in this paragraph is Chai3D. It contains more or less the same functionalities than ReachIn, but this one is an open source project, several haptic researchers are working with it. However, it seems that there is a lack of performance in the context of manipulation with generic virtual environments.

In the next subsections, we describe the haptic device that we use.

B. The Haptic Workstation™

Our haptic peripheral is a product provided by Immersion® Corporation and commercialized under the name of Haptic Workstation™ [3]. This station has the unique advantage of providing tracking and force-feedback for the two hands and fingers, allowing the user to manipulate objects in a very intuitive manner.

The station is made of several subcomponents that are shown in figure 2, namely:

- Two CyberGlove® that tracks the posture of the fingers. For each hand, the 22 joint-angles between phalanxes and metacarpus are measured using built-in constraint gauges allowing us to follow the position of the fingers. The data refresh rate is about $100Hz$.
- A pair of CyberGrasp™ which are exoskeletons that fit around the hands and provide a one-way resistive force-feedback to each fingers. The grasp forces are produced by tendons routed to the fingertips. This peripheral is mainly used to give the feeling that the user is touching or holding an object by blocking the fingers. A force of about $10N$ can be applied separately on each finger.
- A pair of CyberForce® which provide grounded force feedback to the hands and arms. Each can create a force in the three translational directions, modifying the position of the user's wrists if he does not try to resist. However this system is unable to provide a rotational force and thus can not constrain the user to modify the orientation of his hands.
- Two CyberTracks™ that are integrated to the CyberForce® armature. The position and orientation of the wrists are tracked with much higher accuracy and refresh rates than optical and magnetic systems (accuracy of $0.1mm$ in position, 0.1 degree in rotation at a frequency of $1000Hz$).

This device allows a user to interact with his hands. Thus, the haptic library must handle collisions between the hands and the virtual objects of the environment. However, the hands could be considered as deformable objects, and the libraries previously cited in the first subsection do not really address this problem. Moreover, in some cases, the force feedback is applied on several devices at the same time (finger or wrist, or even both hands), depending on the context. These considerations convinced us to create our own haptic rendering framework in order to have a powerful virtual environment manipulation tool.

C. Description of needs

The main task of a haptic rendering engine is to compute force feedback according to the user movements. To allow this, the Mhaptic library should include the following components:

- A **connection** to the Haptic Workstation™. This may seem trivial, but the library has to establish a connection in order to gather input data (hands and fingers positions) and display force feedback.
- A **collision detection** engine: this component determines if, where and when some objects come into contact during the simulation, either with the virtual hands or with other objects. As previously mentioned, our Haptic Workstation™ stimulates the kinesthetic sensory channel. Thus, it is very important to detect all contacts in the VE in order to apply a fast and correct force-feedback to the user.
- A **dynamic** engine for a realistic animation of the objects. This component is used for two purposes. The first one is to realistically animates the objects in order to have a believable manipulation tool. The second one is to ease the force-feedback computation and improve the graphical rendering of the hands. We will deal with this important feature in section III-F.

- A **force-feedback computation** engine: this ensures the correct computations of the force-feedback. This part is a critical point since a wrong computation or an insufficient refresh rate leads to instabilities in the system. As stated in [4], such computation loop must run at least at $500Hz$ in order to be realistic. This constraint is respected in MHaptic.
- A **haptic augmentation tool**: As explained before, most of the existing 3D Scenes available contain only visual data. In order to quickly create haptic manipulation applications, additionnal information should be included.

In the following section, we provide a complete description of MHaptic, including the main force feedback and manipulation techniques and the functionalities.

III. Architecture

In this part, we give an exhaustive overview of every components of the MHaptic library. First, we will provide a brief introduction on the third-party software (subsection III-A) used by MHaptic itself and on the general organization of the library (subsection III-B). Then, we will discuss the first role of MHaptic: to provide a easy and low-level access to the Haptic Workstation™ and its subcomponents. Then, we present the higher level functionalities that have been developed to facilitate the work of the application programmer. In the subsection III-E, we deal with the collision detection system and the dynamic engine that are used to create a physically realistic animated world. And finally, we present the procedure used to allow the two-handed manipulation of virtual objects, and the force-feedback rendering.

A. Third-Party Software

MHaptic is a haptic rendering system. However, combining haptic feedback with visual and audio is almost mandatory. Of course, we did not designed a new graphic engine: we used an existing one that is developed in the laboratory. MVisio is a state of the art computer graphics rendering engine that includes useful features[5]. The most important functionalities are its multi-devices ability (Screen, HMD, CAVE), and the 2D Graphical User Interface tool that allows to easily integrate buttons, windows, etc. in a 3D context (we present how we used GUIs in part III-D2).

Second library used is the AGEIA PhysX™ SDK [6]. It is a commercial software, but free for research. It is a State-of-the-Art physic library that includes continuous collision detection, rigid body dynamics, and deformable objects dynamic like clothes and fluids. It is also usable with a specific hardware (PPU) supposed to increase the computation time. We will deal with it in III-E. We also mention that two other physic libraries already exist: Havok Physics™, which is not freely available, and ODE, an open-source project. ODE was the first physic engine we used, but it appears to be less intuitive and efficient than PhysX, especially because of the loss of performance in large scale environments.

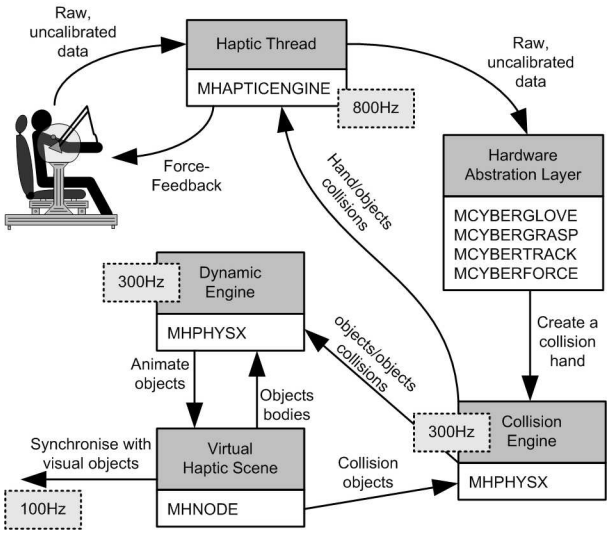

Fig. 3. Organizational Diagram of the MHaptic library.

B. MHaptic Software Organization

As stated in II-C, the role of our library is mainly to enable generic manipulation by the mean of the Haptic Workstation™ into VR applications. The organization diagram shown in figure 3 presents the main modules.

First, directly connected to the Haptic Workstation™ the ***Haptic thread*** is the piece of code that takes care to retrieve data from the two input devices (i.e. the gloves and the trackers) and to send the appropriate repulsion forces to the output devices (i.e. CyberGrasp™ and CyberForce®).

Second, the ***Hardware abstraction layer*** contains the classes that represent and give a direct calibrated access to the devices, easing the programming effort. It contains also the reconstruction of the haptic hand model presented in subsection III-F.

Then, the ***Collision detection system*** uses this hand model to compute the contacts with the objects contained in the ***Virtual Scene***. And finally, the ***Dynamic Engine*** takes care of the realistic animation of virtual objects in order to have a convincing manipulation system.

C. Access to the Haptic Workstation™

The haptic thread objective is to gather and set the values of the Haptic Workstation as fast as possible. Thus, most of the computation should be excluded from this module. However such data is particularly difficult to manipulate. This is the reason why there is an Hardware Abstraction Layer. This module is intended to provide an easy access to the calibrated values/functionnalities of the devices, without decreasing the refresh rate of the haptic thread.

1) Haptic thread: The role of the thread is to ensure that the values are updated as often as possible. In MHaptic, when the 8 devices are connected this thread runs at least at $800Hz$ (the average is around $1000Hz$). Because we use a biprocessor architecture, we put a high priority on this thread in order that it is always executed. Basically it contains two

steps: the first one is to gather input values coming from the CyberTraks™ and CyberGlove®, the second one is to compute and apply the force feedback. Usually, the force feedback is computed according to hand and object contacts. Thus, it is typically a result of the collision detection system: the repulsion force vector is computed as the direction between the penetration point and the collision point. However, if the collision detection update is slow, the force feedback will be also updated slowly, and this is not acceptable because it could result in an unstable system with resonance and vibration [7]. To avoid this problem, the collision detection system send the thread both the collision point and penetration point, not only the repulsion vector. And then, the force vector is thus computed in the haptic thread using always the last updated values. This is the only computation that need to be executed in the haptic thread.

More details about the force feedback computation are given in subsection III-G.

2) Device Calibration: Efficient calibration is one of the most important feature when dealing with the Haptic Workstation™. Except the CyberForce™, all other devices need to be calibrated. In this subsection we will only deal with the dynamic calibration, i.e. the calibration procedure that need to be done every time the device is started, or at every changes of user.

The CyberTrack® are pre-calibrated when the workstation is started. It means that it should be started in a given position. However this position is not easily reproducible perfectly. Thus, we introduce in [8] a procedure to dynamically recalibrate it.

The CyberGrasp™ strings should be calibrated in order that the strings are not pulling too much the fingers or at opposite, completely relaxed.

Finally, the most difficult calibration concerns the CyberGloves™. In [9], authors present a study on different calibration procedures. Our objective is to have a rapid calibration enough precise for purpose of grasping and manipulation. We do not want gesture recognition or other kind of tasks that requires a complex calibration. We know that the raw angle sensors are quite linear. Thus we only need to apply a first order calibration.

$$\begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_{20} \end{bmatrix} = \begin{bmatrix} raw_1 \\ raw_2 \\ \vdots \\ raw_{20} \end{bmatrix} \times \begin{bmatrix} m_1 \\ m_2 \\ \vdots \\ m_{20} \end{bmatrix} + \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_{20} \end{bmatrix} \quad (1)$$

In equation 1, unknowns are p_x and m_x, respectively the offset and scaling parameters. To solve this simple equation, we only need to find two correspondences of α_x and raw_x for each angle. We achieve this by asking the user to place his hand into 4 easily reproducible positions which contain the correspondences (see figure the CyberGlove® calibration window on 4).

Fig. 4. The User Interface integrated with MHaptic

D. High-level Functionalities

In this subsection, we introduce useful functionalities that have been added directly in the library because we found that they were mandatory in most of the applications using the Haptic Workstation™.

1) User Comfort Improvement: Our first experience with the Haptic Workstation™ has shown that this device is uncomfortable to use during long sessions especially when dealing with manipulation. The main reason is the uncomfortable posture of the arms, which must be kept outstretched horizontally while supporting the weight of an exoskeleton. The exoskeleton has mechanical counterweights, but in some position they are not enough heavy. To prevent this situation, we have integrated a functionality that uses the CyberForce™ force feedback to remove the weight of the exoskeleton. More details can be found in [8].

2) Graphical User Interface: MVisio allows for creating 2D GUI integrated with the 3D environment. Several features of MHaptic, particularly devices connection, calibration testing and debugging information could be displayed on a user interface using a single C++ function. This is really useful because it avoids to the application developer to manage this himself, and thus, speeds up the creation process. (see figure 4)

3) Workspace Extension: When immersed into a large scale virtual environment, the user is not able to touch every objects. A metaphor should be used to move him into the VE. In [10], authors present an evaluation of 3 different techniques to achieve the interaction with objects that appear bigger than the workspace of the haptic device. We used a method similar to the bubble technique [11]. When the user is moving his arms near the limit of workspace, he enters into an area that displace the virtual camera: arms to the front moves the camera forward, arms to one side turn the camera. In addition, force feedback is applied to prevent that the user enter this area, and a very smooth visual fog provides visual warning and allow the user to understand why he is moving and feeling something that he do not see usually.

E. Realistically Animated Virtual Environment

In this part, we describe one of the main component of the system, i.e. the dynamic engine and collision detection system.

As previously mentioned in III-A, we have used existing third party software from the PhysX library.

1) The Collision Detection System: works with geometries. Every "touchable" objects in the virtual scene has to be created using PhysX geometries. The hands are also composed by geometries. Two geometries can be passed as argument to the collision detection system which will return many information including collision points (position in local or global coordinate system) and penetration distance.

2) The Dynamic Animation Engine: is the second main component of the PhysX library. Its role is to move a set of rigid bodies (the animation entities) in the world by a physically realistic manner. A rigid body is parameterized by its weight, its center of gravity and its inertia tensor. Bodies could also be linked together by joints (spring, distance, hinge, etc.). And finally, a body is affected by all the forces applied on it, resulting from gravity, joints, and collisions (when a collision is detected between two geometries, a correction force is applied on the associated bodies, in order to prevent the penetration).

3) A Haptic Node: is a touchable object of the MHaptic framework. We can distinguish two kind of nodes, called dynamic or static. A static object can not be moved. For example, in a virtual flat example, the floor, the walls, the bed and every "big" objects could not be moved. On opposite, the "small" objects are usually dynamic, and thus could be manipulated by the user. Both static and dynamic objects contains geometries and are registered in the collision detection system, but only the dynamic objects have a body. In order to speed up the distribution, reuse and editing of haptic nodes, we have created XML serialization and deserialization[1] functions using the libxml library.

F. Spring Damper Hands

When using the Haptic Workstation™, the interaction "tools" are the hands themselves. It means that a virtual model of the hands has to be created. Several approaches have been primarily tested based on different methodologies. This is presented in the first subsection, and the second subsection deals with the implementation of the chosen method.

1) Problem Statement: Two approaches to perform touchable virtual environments could be used:

- One consists in positioning a virtual interaction point at the same position than the device itself (hard link). The repulsion force is then computed to be proportional to the penetration distance into a virtual object. This method is relatively easy to implement and works well with simple scenarios. However, it results also into visual inconsistencies because the user could force the device to stay into a virtual object even if force feedback is applied. This causes a break in presence [12].
- Second technic consists in using a "ghost-object" [13]. A ghost object is weakly linked to the position of the device. It cannot penetrate into virtual objects. With this method, the repulsion force is proportional to the difference between ghost-object position and device position. This approach is more generic, works usually well in all kind of situations and solves some of the drawbacks of the previous method. But it induces that sometimes, when the user is penetrating into an object, the position of the ghost-object is not valid, i.e. not in the position of the real hand.

[1]The Document Type Definition (DTD) of the XML file is available on http://vrlab.epfl.ch/~reno/MHAPTIC/Haptic2Scene.dtd

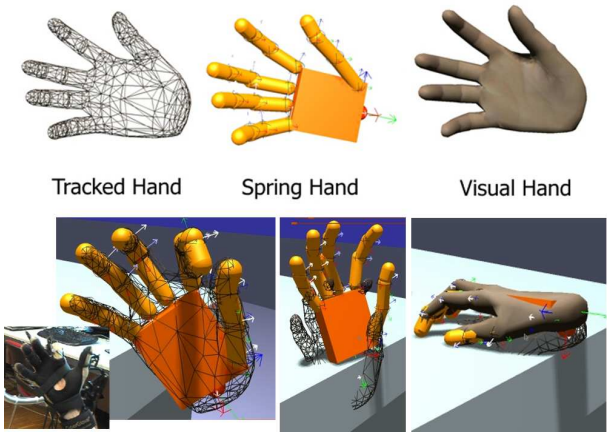

Fig. 5. The "three hands".

Both approaches have advantages and drawbacks. First one provides visual inconsistencies, second one gives proprioceptive discrepancies. In [14], authors investigate which of the two methods generates less breaks in presence for the user, and they conclude that "Users are more sensitive to visual interpenetration than to visual-proprioceptive discrepancy".

Thus, it appears to be better in term of presence to use the second method. But if this method is quite easy to implement using a Phantom®-like haptic device which interaction paradigm is point-based, it is more difficult when dealing with two-handed interaction. In [15] or in [16], authors shows that using a kind of mass-spring system linking the god-object to the real position of the haptic device is a promising technique. In [17] and [18], Borst et Al. extend the mass-spring system to a whole hand. Starting from this last article, we created our mass-spring-damper system for the two hands using the PhysX tools.

2) Implementation: Within our library we could distinguish three different hand models as shown on figure 5. First, the **tracked-hand** reflects the real hand using the values returned directly from the Haptic Workstation™ after calibration. The posture and position of this hand matches the reality. The second hand is the **spring-hand**. This one is composed by several rigid bodies (cylinders for the phalanxes, and a box for the palm). Finally, the **visual-hand** is, of course, the one displayed to the user, and matches exactly the posture/position of the spring-hand.

The phalanxes rigid bodies of the spring-hand are linked together using PhysX spherical joints. These joints are enough

parameterizable to be constraint on a single axis or even using angle limits. We use this functionality to avoid that the spring-hand takes physiologically impossible postures. Then, on each joint we attach a PhysX motor. A motor applies a force on a joint to make it reach a certain position. Therefore, at each time step, we check the angle between two real phalanxes and use the motor associated to them to set the same angle of the spring-hand. The advantage of using a motor is that it sets a force on a rigid body. And this force could be propagated along all the phalanxes, even until the wrist. This mechanism allows for setting the posture of the spring hand. Concerning the position of the spring-hand, we used a different procedure. The PhysX library provides also kinematic actors. A kinematic actor is an object that is not moved directly by the dynamic engine, but by the user via a function. Thus, we have created a kinematic actor controlled by the CyberTrack^{TM}, and linked our spring-hand to this actor using a spring joint. This results in a spring-hand which always takes the same rotation than the real hand, and that tries to reach the same position (if it is possible).

The parameters of the spring-hand are the elasticity, damping constant of every joints, and the weight (inertia) of the rigid bodies of the phalanxes. We will discuss how to choose them in section V.

The spring-hand has also a great advantage: it considerably simplify the computation of the repulsion forces, and the simulation of grasping as we present it in the next subsection.

G. Force Feedback Rendering

The main goal of a haptic library is the simulation of the sense of touch, and in our case, the proprioception. Thus, forces have to be computed realistically in order that the user apprehend well the VE.

In addition to visual consistency, using a ghost-hand present another advantage: it is straightforward to compute the force feedback. In fact, the difference of posture between the ghost hand and the tracked hand represents the forces that have to be applied. As shown on figure 6, where the yellow arrows represent wrist force feedback, force feedback is generated when the user touches an object and makes a pressure on it. When looking at the upper right picture, we can see that if the user resists to the force feedback applied on his fingers, it will be propagated until the wrist. This is also extended to 2 hands manipulation: force will be propagated to the hand resisting the less.

Finally, in every collision detection and dynamic animation loops (around $300Hz$), the position to reach for each fingertip and wrist is sent to the haptic thread that will handle itself the computation of the force feedback using the last updated values (around $800Hz$). This ensure the smoothness of the haptic simulation.

In this section, we have presented the implementation of MHaptic, describing its organization, its functionalities, and its rendering techniques used to provide to the user a convincing haptic experience. In next section we will deal with the integration of MHaptic into existing virtual environments.

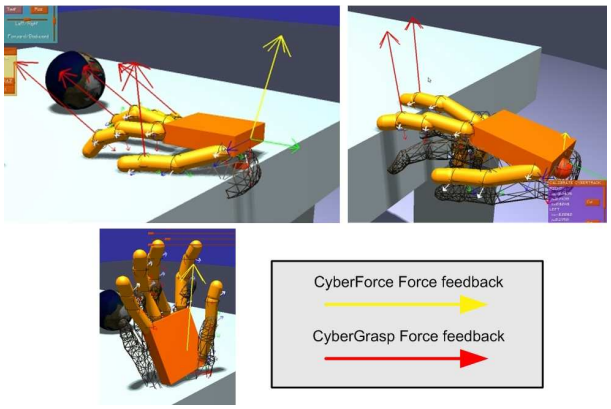

Fig. 6. The computation of force feedback uses the tracked hand position (wireframe) and the spring-hand position (solid).

IV. The Haptic Scene Creator

MHaptic allows to quickly set up a working system that includes haptic feedback and manipulation of virtual objects. However, the virtual environments created by 3D designers includes rarely the haptic information needed by the haptic libraries. Starting from this consideration, we have created a complete application whose goal is to allow anyone to easily edit a 3D scene in order to "augment" the visual objects with *haptic information*.

Haptic information refers to the geometries that approximates a visual object. A good-looking visual mesh is by far too much complex for the collision detection system. Moreover, concerning dynamic nodes, *haptic information* also refers to weight, inertia tensor, and center of gravity of the node.

A. Basic functionalities

The Haptic Scene Creator (HSC) interface is inspired by usual 3D design software like Maya or 3DSMax (see figure 7). It is implemented using MHaptic itself and thus use MVisio as graphical rendering library. This implies that the 3D scenes that can be load have been previously imported to the MVisio `.MVE` file format [5]. But MVisio uses models from Autodesk® 3DSMax, so every 3DSMax scenes could be haptically augmented. We remind that the result of the augmentation of a scene is a human-readable XML file.

Basically, HSC allows to open a visual scene, to navigate into it, and to select an object. When an object is selected,

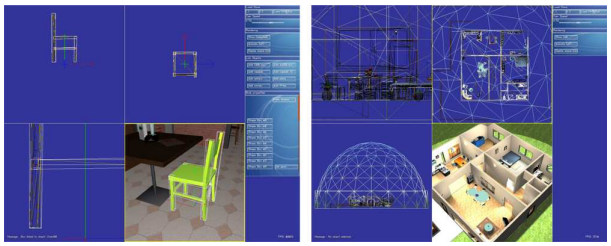

Fig. 7. The Haptic Scene Creator interface.

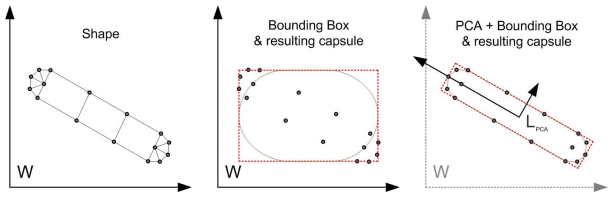

Fig. 8. The PCA simplifies creation of geometries

it is displayed on the four windows. Then, it is possible to select some of the vertices of the object, and to create and parameterize the geometries approximating it (using the simple primitives, box, sphere, capsule, or the complex one like convex object or penetration maps). Geometries also have a texture (described by friction parameters and bounciness). Moreover, if an object is dynamic, the weight has to be entered. Then, one can animate the scene and send balls on the objects to check their behavior. Finally it is possible to save the XML haptic file to load it after into any application using MHaptic.

Even if it is better than manual editing of the XML file, in this state, the HSC suffers from a lack of flexibility. At usage, it is not possible to quickly augment a complete scene. Especially because it appears to be difficult to parameterize the size of the boxes, and to place them well to approximate a virtual object. Thus, we added many functionalities presented in the next subsection, making of the HSC a really powerful tool.

B. Advanced functionalities

In order to speed up the parameterizing of geometries, we perform a Principal Component Analysis (PCA) on the cloud of points that the user wants to approximate [19]. In our case, the role of PCA is to extract the orientation of the cloud of vertices, in order to find the best coordinate system. Having this coordinate system, we compute the bounding box to find the parameters of the geometry. The figure 8 illustrates this mechanism. User needs only to select a part of an object (or the whole object), and to click on the geometry which best matches the general shape. In 95% of the cases this simple action is sufficient, but sometimes, the user has to rotate or translate a bit the geometry: translation is achieved easily by moving object with the mouse, and rotation is made using an arcball, the most intuitive way to rotate a 3D object using a 2D device [20]. Copy-Pasting is also possible for objects. This is very useful when dealing with scenes that include many time the same 3D model. And finally, our interface also allows the creation of convex shapes, and of penetration maps as shown on figure 9. In next subsection, we present an evaluation of the HSC.

C. Usability

As shown previously, HSC includes many features aiming at simplifying the task of a Haptic application Designer. Our test scene is a complex scene representing a 4-rooms flat (kitchen, bathroom, bed, office and living-room). This scene contains 398 visual nodes. Main goal is to use the HSC to augment every touchable object with a good level of approximation in order to manipulate objects easily. This task took almost 3 hours, time to create 612 geometries for 167 static objects and 114 dynamic objects. We do not have performed the same procedure on the full scene without using the advanced functionalities (and neither using a simple text editor to write by hand the XML file). But, to give an idea of the improvements, augmenting a chair without advanced functionalities took 5 (tedious!) minutes, whereas it took around 1 minute using the PCA.

Thus, the HSC appears to be an essential tool to easily create virtual environment that are touchable and manipulable. This is mandatory when dealing with complex haptic applications, because less time is spent on the creation of the environment, the more time it gives to the application itself.

V. Performance and Applications

Our haptic library provides many tools that allows to quickly and easily create a complete haptic application. We will not deal too much about coding, however we have to mention that when using MHaptic one hundred lines of C++ code are enough to allow a user to have a complete interactive visual haptic VE. This includes also the interface to connect and calibrate the hardware.

The most impressive feature of MHaptic is the mass-spring-damper system for controlling the hands. The main problem is the parametrization of the spring, damping and mass values. After trying to put the same weight of phalanxes than in the reality, we face a strong resonance problem forcing us to put a greater mass to the first phalanx (the one near the palm) and a smaller until fingertip. Then, to choose spring and damper, we notice that increasing too much the damping factor or decreasing the spring factor induces the same effect: the lag. Phalanxes moves indeed slowly and do not react quickly to rapid user movements. By opposition, increasing too much spring factor or decreasing damping tends to create resonance. The phalanxes are vibrating and resulting force feedback is really unpleasant. Taking into account these considerations, it is not really hard to find the good values. We use the same values for every springs of the hand.

The force-feedback on the fingers is also convincing. Sliding a finger on the border of a table until the contact is lost gives a

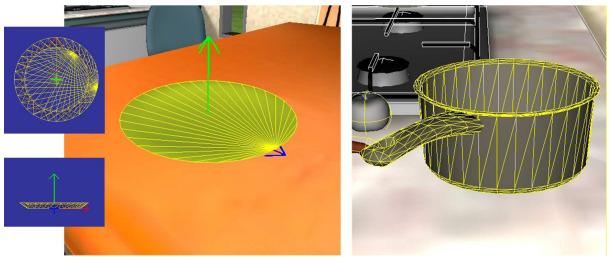

Fig. 9. Convex shape on the left, and penetration map on the right.

very realistic feeling of the edge. The forces are unfortunately limited to one direction due to the CyberGrasp™ device. However this method has some limits. When a strong distance is observed between the tracked hand and the spring hand (this may happen when the user collides a static object like a wall but forces the movement too far despite the force-feedback), the force applied on the base of the spring-hand becomes very large. As a result the rigid bodies of the spring-hand are compressed on the surface of the static object and the simulation becomes unstable due to the huge opposite forces (spring force versus resting forces). Similar effects are observed when the fingers of the tracked-hand penetrate too deeply into a grasped object.

Concerning the grasping and the manipulation, we achieve good results with objects composed of basic primitives. However, we experienced sometimes difficulties with the dynamic objects approximated by penetration maps: they can get blocked when penetrating into the hand.

MHaptic is thus a library that allows for generic haptic manipulation. There is no need for creating specific force-feedback by examining which objects are colliding. Everything is computed and a haptic application programmer do not need to handle haptic rendering (but this does not mean that he can not handle it himself). This is particularly useful when focusing on manipulation training, assembly, virtual environment exploration, and so on. We already made a mixed-reality assembly training application that use most of the features of MHaptic [21], but also truck gearbox simulation, or a haptic juggling system.

VI. Conclusion

In this paper we have presented an overview of the main existing haptic libraries. We evaluate them in order to check if they can fit our specific needs: allowing a two handed haptic feedback using Haptic Workstation™ within generic virtual environments. This evaluation convince us to develop new tools, that we grouped in the MHaptic framework. MHaptic is a multi-threaded haptic rendering system for touching, grasping and manipulation of virtual objects. This library uses the state-of-the-art Ageia PhysX physic engine to drive the objects and to create our spring hand model. This model is a convincing technique that has the advantage to avoid visual breaks in presence by preventing fingers to penetrate into virtual objects. Moreover when used with good parameters, it allowed us to have a generic manipulation system working in most cases. Because most of the 3D models available do not integrate haptic or even physical information, we also present a 3DSMax-like software intended to quickly and intuitively augment 3D scenes. With this tool, it is now possible to parameterize virtual scenes to include the weight, the center of gravity or the approximating geometries used by collision detection.

Thus, MHaptic provides a truly generic haptic rendering model allowing manipulation of 3D objects with two hands. In the future, we plan to integrate texture rendering to improve the system and to validate the approach used in the case of two-handed manipulation of the same object.

References

[1] ReachIn API 4.2, by ReachIn Technologies. [Online]. Available: http://www.reachin.se/products/ReachinAPI/

[2] F. Conti, F. Barbagli, D. Morris, and C. Sewell, "Chai: An open-source library for the rapid development of haptic scenes," in *Demo paper presented at IEEE World Haptics*, March 2005.

[3] Immersion® Corporation, Haptic Workstation™. [Online]. Available: http://www.immersion.com/3d/products/haptic_workstation.php

[4] P. Fuchs and G. Moreau, *Le Traité de la Réalité Virtuelle, deuxième édition*. Les Presses de l'Ecole des Mines de Paris, 2004, vol. 1.

[5] A. Peternier, D. Thalmann, and F. Vexo, "Mental vision: a computer graphics teaching platform," in *proceedings of the 2006 Edutainment Conference*, 2006, pp. 223–232.

[6] PhysX™ by AGEIA. [Online]. Available: http://www.ageia.com/physx/index.html

[7] D. C. Ruspini, K. Kolarov, and O. Khatib, "The haptic display of complex graphical environments," in *proceedings of the 24th annual conference on Computer graphics and interactive techniques : SIGGRAPH'97*. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 1997, pp. 345–352.

[8] R. Ott, M. Gutierrez, D. Thalmann, and F. Vexo, "Improving user comfort in haptic virtual environments trough gravity compensation," in *proceedings of the First Joint Eurohaptics Conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems (WORLDHAPTICS'05)*, 2005, pp. 401–409.

[9] G. D. Kessler, L. F. Hodges, and N. Walker, "Evaluation of the cyberglove as a whole-hand input device," *ACM Trans. Comput.-Hum. Interact.*, vol. 2, no. 4, pp. 263–283, 1995.

[10] L. Dominjon, A. Lcuyer, J. Burkhardt, and S. Richir, "A comparison of three techniques to interact in large virtual environments using haptic devices with limited workspace," *Lecture Notes in Computer Science, Springer Verlag*, vol. 4035, pp. 288–299, 2006.

[11] L. Dominjon, A. Lcuyer, J. Burkhardt, G. Andrade-Barroso, and S. Richir, "The bubble technique: Interacting with large virtual environments using haptic devices with limited workspace," in *proceedings of the First Joint Eurohaptics Conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems (WORLDHAPTICS'05)*.

[12] M. Slater and A. Steed, "A virtual presence counter," *Presence : Teleoperators and Virtual Environments*, vol. 9, no. 5, pp. 413–434, October 2000.

[13] C. B. Zilles and J. K. Salisbury, "A constraint-based god-object method for haptic display," in *proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 3, 1995, pp. 146–151.

[14] E. Burns, S. Razzaque, A. T. Panter, M. C. Whitton, M. R. McCallus, and J. Frederick P. Brooks, "The hand is more easily fooled than the eye: Users are more sensitive to visual interpenetration than to visual proprioceptive discrepancy," *Presence : Teleoperators and Virtual Environments*, vol. 15, no. 1, pp. 1–15, February 2006.

[15] P. Mitra and G. Niemeyer, "Dynamic proxy objects in haptic simulations," in *proceedings of the IEEE Conference on Robotics, Automation and Mechatronics*, vol. 2, 2004, pp. 1054–1059.

[16] F. Conti, O. Khatib, and C. Baur, "Interactive rendering of deformable objects based on a filling sphere modeling approach," in *proceedings of the IEEE International Conference on Robotics and Automation*, 2003, pp. 3716–3721.

[17] C. W. Borst and A. P. Indugula, "Realistic virtual grasping," in *proceedings of the 2005 IEEE Conference on Virtual Reality (VR'05)*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 91–98, 320.

[18] ——, "A spring model for whole-hand virtual grasping," *Presence : Teleoperators and Virtual Environments*, vol. 15, no. 1, pp. 47–61, February 2006.

[19] I. T. Jolliffe, *Principal Component Analysis*. Springer, 2002.

[20] K. Shoemake, "Arcball: a user interface for specifying three-dimensional orientation using a mouse," in *proceedings of the conference on Graphics Interface '92*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1992, pp. 151–156.

[21] R. Ott, D. Thalmann, and F. Vexo, "Haptic feedback in mixed-reality environment," in *proceedings of the 25th Computer Graphics International Conference (CGI'2007)*, 2007.